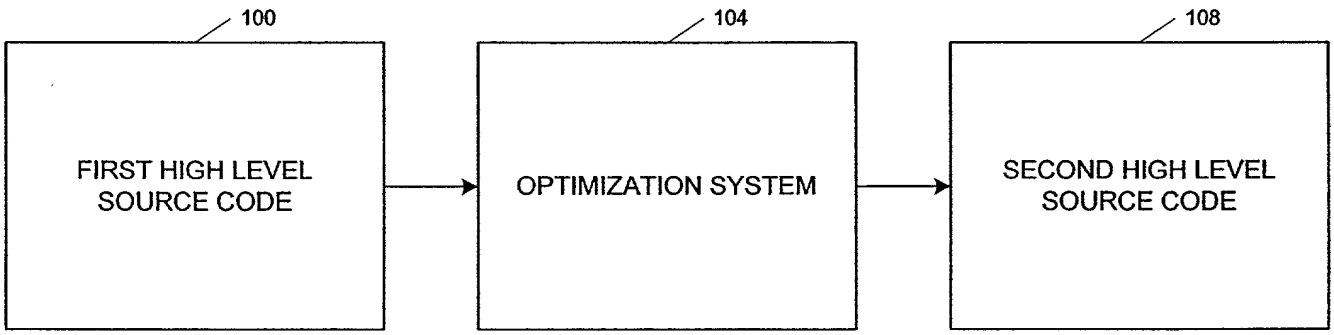
100
FIRST HIGH LEVEL SOURCE CODE
104
OPTIMIZATION SYSTEM
108
SECOND HIGH LEVEL SOURCE CODE

FIG. 1

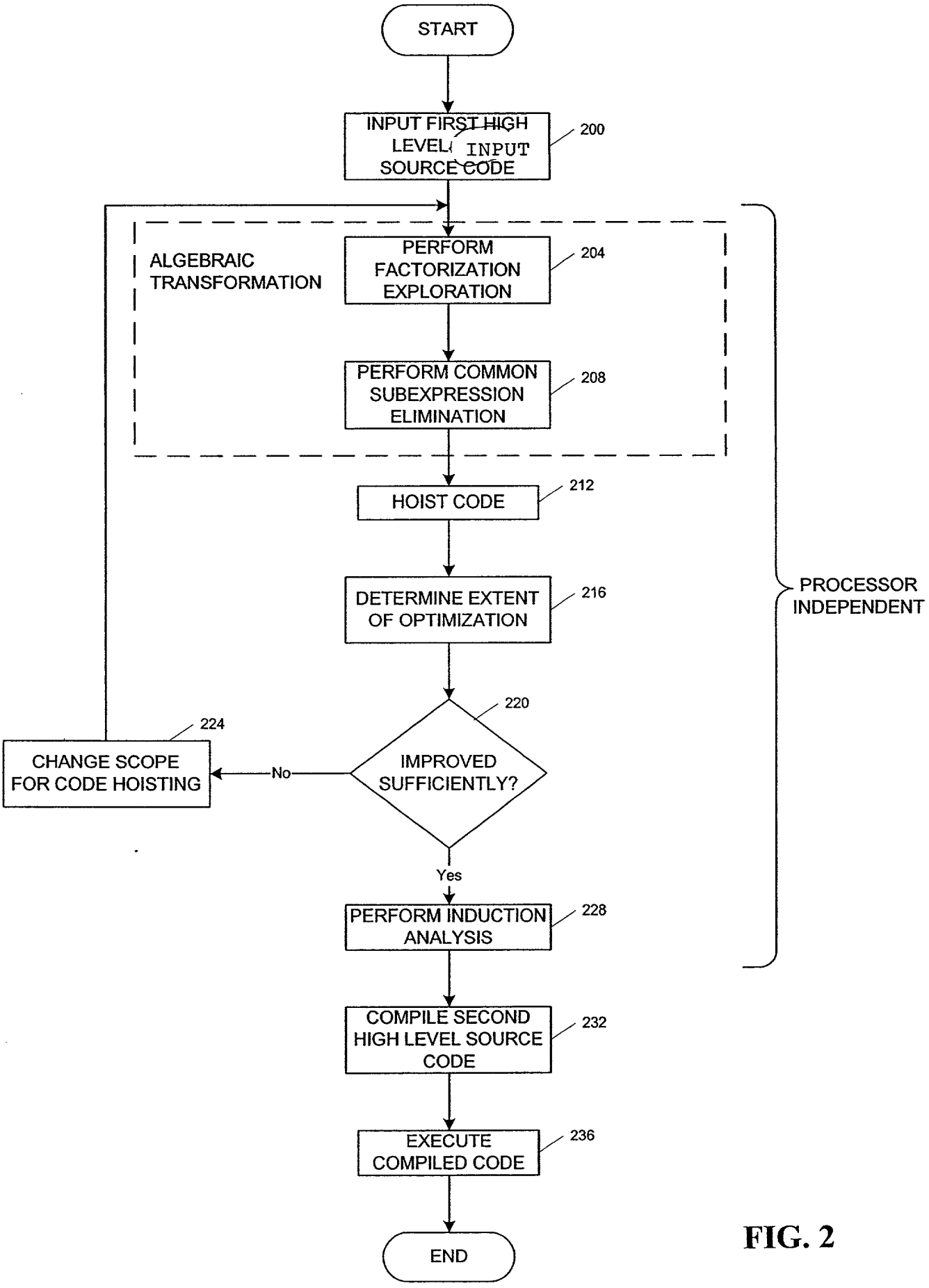
START
INPUT FIRST HIGH LEVEL INPUT SOURCE CODE
200
ALGEBRAIC TRANSFORMATION
PERFORM FACTORIZATION EXPLORATION
204
PERFORM COMMON SUBEXPRESSION ELIMINATION
208
HOIST CODE
212
DETERMINE EXTENT OF OPTIMIZATION
216
IMPROVED SUFFICIENTLY?
220
No
CHANGE SCOPE FOR CODE HOISTING
224
Yes
PERFORM INDUCTION ANALYSIS
228
COMPILE SECOND HIGH LEVEL SOURCE CODE
232
EXECUTE COMPILED CODE
236
END
PROCESSOR INDEPENDENT

FIG. 2

```
for (y=0; y<M+3; ++y) {
  for (x=0; x<N+5; ++x) {
    ...
    if ((x-3)>=1 && (x-5)<=N-2 && (y-2)>=1 && (y-3)<=M-2) {
      if ((x-5)>=1 && (y-3)>=1) {
        if (out_compute == 255) {
          if (comp_edge_pixels[((x-4)%3)*3+(y-2)%3]<comp_edge_middle) out_compute=0;
                                ...
          if (comp_edge_pixels[((x-4)%3)*3+(y-4)%3]<comp_edge_middle) out_compute=0;
                                ...
          if (comp_edge_pixels[((x-5)%3)*3+(y-4)%3]<comp_edge_middle) out_compute=0;
                                ...
        }
      }
      if ((x-3)<=N-2 && (y-2)<=(M-2)) {
        maxdiff_compute =
          max13(abs(gauss_xy_pixels[((x-2)%3)*3+(y-1)%3]
                    - gauss_xy_middle), maxdiff_compute);
                                ...
        maxdiff_compute =
          max13(abs(gauss_xy_pixels[((x-2)%3)*3+(y-3)%3]
                    - gauss_xy_middle), maxdiff_compute);
                                ...
        maxdiff_compute =
          max13(abs(gauss_xy_pixels[((x-3)%3)*3+(y-3)%3]
                    - gauss_xy_middle), maxdiff_compute);
                                ...
      }
    }
    ...
  }
}
```

FIG. 3

```
for (y=0; y<M+3; ++y) {
  for (x=0; x<N+5; ++x) {
    ...
   if(x>=4 && x<=N+3 && y>=3 && y<=M+1) {
     if((x-5)>=1 && (y-3)>=1) {
      if (out_compute == 255) {
        csexmin4mod3x3 = ((x-4)%3)*3;
        cseymin4mod3 = (y-4)%3;
        if(comp_edge_pixels[csexmin4mod3x3 +(y-2)%3]<comp_edge_middle) out_compute=0;
                        ...
        if(comp_edge_pixels[csexmin4mod3x3 +cseymin4mod3]<comp_edge_middle) out_compute=0;
                        ...
        if(comp_edge_pixels[((x-5)%3)*3 +cseymin4mod3]<comp_edge_middle) out_compute=0;
                        ...
      }
     }
     if ((x-3)<=N-2 && (y-2)<=(M-2)) {
       csexmin2mod3x3 = ((x-2)%3)*3;
       cseymod3 = y%3;  /* = (y-3)%3  */
       maxdiff_compute =
         max13(abs(gauss_xy_pixels[csexmin2mod3x3 + (y-1)%3]
                   - gauss_xy_middle), maxdiff_compute);
                     ...
       maxdiff_compute =
         max13(abs(gauss_xy_pixels[csexmin2mod3x3 + cseymod3]
                   - gauss_xy_middle), maxdiff_compute);
                     ...
       maxdiff_compute =
         max13(abs(gaus_xy_pixels[(x%3)*3 + cseymod3]
                   - gauss_xy_middle), maxdiff_compute);
                     ...
     }
    }
    ...
  }
}
```

FIG. 4

```
distributivity:      (x + 4)%3 = (x%3 + 4%3)%3
constant folding:              = (x%3 + 1)%3
constant unfolding:            = (x%3 + 1%3)%3
invert distributivity:         = (x + 1)%3
                       (a)

modulo expansion: (x+2)%3 = 3 - x%3 - (x+1)%3

                       (b)
```

FIG. 5

Downloaded from http://www.patentdocs.org/

```
for (y=0; y<M+3; ++y) {
  cseymod3 = y%3;
  cseymin1mod3 = (y-1)%3;
  cseymin2mod3 = (y-2)%3;
  cseymin4mod3 = (y-4)%3;
  for (x=0; x<N+5; ++x) {
    ...
   if(x>=4 && x<=N+3 && y>=3 && y<=M+1) {
     if((x-5)>=1 && (y-3)>=1) {
      if (out_compute == 255) {
        csexmin4mod3x3 = ((x-4)%3)*3;
        if(comp_edge_pixels[csexmin4mod3x3 +cseymin2mod3]<comp_edge_middle) out_compute=0;
                    ...
        if(comp_edge_pixels[csexmin4mod3x3 +cseymin4mod3]<comp_edge_middle) out_compute=0;
                    ...
        if(comp_edge_pixels[((x-5)%3)*3 +cseymin4mod3]<comp_edge_middle) out_compute=0;
                    ...
      }
     }
     if ((x-3)<=N-2 && (y-2)<=(M-2)) {
       csexmin2mod3x3 = ((x-2)%3)*3;
       maxdiff_compute =
         max13(abs(gauss_xy_pixels[csexmin2mod3x3 + cseymin1mod3]
                   - gauss_xy_middle), maxdiff_compute);
                    ...
       maxdiff_compute =
         max13(abs(gauss_xy_pixels[csexmin2mod3x3 + cseymod3]
                   - gauss_xy_middle), maxdiff_compute);
                    ...
       maxdiff_compute =
         max13(abs(gaus_xy_pixels[(x%3)*3 + cseymod3]
                   - gauss_xy_middle), maxdiff_compute);
                    ...
     }
    }
    ...
  }
}
```

FIG. 6

```
cseymod3 = -1;
for (y=0; y<M+3; ++y) {
  cseymin1mod3 = cseymod3;
  cseymod3 = y%3;
  cseymin2mod3 = 3-cseymod3-cseymin1mod3;
  for (x=0; x<N+5; ++x) {
    ...
   if(x>=4 && x<=N+3 && y>=3 && y<=M+1) {
     if((x-5)>=1 && (y-3)>=1) {
      if (out_compute == 255) {
        csexmin4mod3x3 = ((x-4)%3)*3;
        if(comp_edge_pixels[csexmin4mod3x3 +cseymin2mod3]<comp_edge_middle) out_compute=0;
                        ...
        if(comp_edge_pixels[csexmin4mod3x3 +cseymin1mod3]<comp_edge_middle) out_compute=0;
                        ...
        if(comp_edge_pixels[((x-5)%3)*3 +cseymin1mod3]<comp_edge_middle) out_compute=0;
                        ...
      }
     }
     if ((x-3)<=N-2 && (y-2)<=(M-2)) {
       csexmin2mod3x3 = ((x-2)%3)*3;
       maxdiff_compute =
         max13(abs(gauss_xy_pixels[csexmin2mod3x3 + cseymin1mod3]
                   - gauss_xy_middle), maxdiff_compute);
                     ...
       maxdiff_compute =
         max13(abs(gauss_xy_pixels[csexmin2mod3x3 + cseymod3]
                   - gauss_xy_middle), maxdiff_compute);
                     ...
       maxdiff_compute =
         max13(abs(gaus_xy_pixels[(x%3)*3 + cseymod3]
                   - gauss_xy_middle), maxdiff_compute);
                     ...
     }
    }
    ...
  }
}
```

FIG. 7

```
cseymod3 = -1;
for (y=0; y<M+3; ++y) {
  cseymin1mod3 = cseymod3;
  cseymod3 = y%3;
  cseymin2mod3 = 3-cseymod3-cseymin1mod3;
  for (x=0; x<N+5; ++x) {
     ...
   if(x>=4 && x<=N+3 && y>=3 && y<=M+1) {
     csexmod3x3 = (x%3)*3;
     csexmin2mod3x3 = ((x-2)%3)*3;
     csexmin4mod3x3 = ((x-4)%3)*3;
     csexmin5mod3x3 = ((x-5)%3)*3;
     if((x-5)>=1 && (y-3)>=1) {
      if (out_compute == 255) {
        if(comp_edge_pixels[csexmin4mod3x3 +cseymin2mod3]<comp_edge_middle) out_compute=0;
                        ...
        if(comp_edge_pixels[csexmin4mod3x3 +cseymin1mod3]<comp_edge_middle) out_compute=0;
                        ...
        if(comp_edge_pixels[csexmin5mod3x3 +cseymin1mod3]<comp_edge_middle) out_compute=0;
                        ...
      }
     }
     if ((x-3)<=N-2 && (y-2)<=(M-2)) {
       maxdiff_compute =
         max13(abs(gauss_xy_pixels[csexmin2mod3x3 + cseymin1mod3]
                   - gauss_xy_middle), maxdiff_compute);
                        ...
       maxdiff_compute =
         max13(abs(gauss_xy_pixels[csexmin2mod3x3 + cseymod3]
                   - gauss_xy_middle), maxdiff_compute);
                        ...
       maxdiff_compute =
         max13(abs(gaus_xy_pixels[csexmod3x3 + cseymod3]
                   - gauss_xy_middle), maxdiff_compute);
                        ...
     }
    }
    ...
  }
}
```

FIG. 8

```
cseymod3 = -1;
for (y=0; y<M+3; ++y) {
  cseymin1mod3 = cseymod3;
  cseymod3 = y%3;
  cseymin2mod3 = 3-cseymod3-cseymin1mod3;
  for (x=0; x<N+5; ++x) {
   csexmod3x3 = (x%3)*3;
   csexmin2mod3x3 = ((x-2)%3)*3;
   csexmin4mod3x3 = ((x-4)%3)*3;
   csexmin5mod3x3 = ((x-5)%3)*3;
     ...
   if(x>=4 && x<=N+3 && y>=3 && y<=M+1) {
     if((x-5)>=1 && (y-3)>=1) {
      if (out_compute == 255) {
        if(comp_edge_pixels[csexmin4mod3x3 +cseymin2mod3]<comp_edge_middle) out_compute=0;
                        ...
        if(comp_edge_pixels[csexmin4mod3x3 +cseymin1mod3]<comp_edge_middle) out_compute=0;
                        ...
        if(comp_edge_pixels[csexmin5mod3x3 +cseymin1mod3]<comp_edge_middle) out_compute=0;
                        ...
      }
     }
     if ((x-3)<=N-2 && (y-2)<=(M-2)) {
       maxdiff_compute =
         max13(abs(gauss_xy_pixels[csexmin2mod3x3 + cseymin1mod3]
                   - gauss_xy_middle), maxdiff_compute);
                        ...
       maxdiff_compute =
         max13(abs(gauss_xy_pixels[csexmin2mod3x3 + cseymod3]
                   - gauss_xy_middle), maxdiff_compute);
                        ...
       maxdiff_compute =
         max13(abs(gaus_xy_pixels[csexmod3x3 + cseymod3]
                   - gauss_xy_middle), maxdiff_compute);
                        ...
     }
    }
    ...
  }
}
```

FIG. 9

```
cseymod3 = -1;
for (y=0; y<M+3; ++y) {
  cseymin1mod3 = cseymod3;
  cseymod3 = y%3;
  cseymin2mod3 = 3-cseymod3-cseymin1mod3;
  csexmod3x3=-3;
  for (x=0; x<N+5; ++x) {
    csexmin1mod3x3 = csexmod3x3;
    csexmod3x3 = (x%3)*3;
    csexmin2mod3x3 = 9-csexmod3x3-csexmin1mod3x3;
      ...
    if(x>=4 && x<=N+3 && y>=3 && y<=M+1) {
     if((x-5)>=1 && (y-3)>=1) {
      if(out_compute == 255) {
        if(comp_edge_pixels[csexmin1mod3x3 +cseymin2mod3]<comp_edge_middle) out_compute=0;
                          ...
        if(comp_edge_pixels[csexmin1mod3x3 +cseymin1mod3]<comp_edge_middle) out_compute=0;
                          ...
        if(comp_edge_pixels[csexmin2mod3x3 +cseymin1mod3]<comp_edge_middle) out_compute=0;
                          ...
      }
     }
     if ((x-3)<=N-2 && (y-2)<=(M-2)) {
       maxdiff_compute =
         max13(abs(gauss_xy_pixels[csexmin2mod3x3 + cseymin1mod3]
                   - gauss_xy_middle), maxdiff_compute);
                     ...
       maxdiff_compute =
         max13(abs(gauss_xy_pixels[csexmin2mod3x3 + cseymod3]
                   - gauss_xy_middle), maxdiff_compute);
                     ...
       maxdiff_compute =
         max13(abs(gaus_xy_pixels[csexmod3x3 + cseymod3]
                   - gauss_xy_middle), maxdiff_compute);
                     ...
     }
    }
    ...
  }
}
```

FIG. 10

```
cseymod3 = -1;
for (y=0; y<M+3; ++y) {
  cseymin1mod3 = cseymod3;
  cseymod3 = y%3;
  cseymin2mod3 = 3-cseymod3-cseymin1mod3;
  csexmod3x3=-3;
  cseymin1mod2 = (y-1)%2;
  cseymod2=1-cseymin1mod2;
  for (x=0; x<N+5; ++x) {
    csexmin1mod3x3 = csexmod3x3;
    csexmod3x3 = (x%3)*3;
    csexmin2mod3x3 = 9-csexmod3x3-csexmin1mod3x3;
    csexmin1x2 = (x-1)*2;
    csexmin3x2 = csexmin1x2-4;
                 ...
    if (x>=3 && x<N+3 && y>=2 && y<M+2)
      tmparray[(csexmin3x2+ cseymod2)%160
                          + (csexmin3x2+ cseymod2)/160*256 + 96]
        = comp_edge_pixels[csexmod3x3
                                + cseymin2mod3] = maxdiff_compute;
                 ...
    if (x>= 1 && x<N+1 && y>=1 && y<=M)
      tmparray[(csexmin1x2 + cseymin1mod2)%64
                          + (csexmin1x2 + cseymin1mod2)/64*256]
        = gauss_xy_pixels[csexmin1mod3x3
                                + cseymin1mod3] = gauss_xy_compute;

    ...
  }
}
```

FIG. 11

```
cseymod3=-1;
for (y=0; y<M+3; ++y) {
  cseymin1mod3=cseymod3;
  cseymod3 ++;
  if(cseymod3 >= 3){cseymod3 -= 3;}
  cseymin2mod3 = 3-cseymod3-cseymin1mod3;
  cseymin1mod2 = (y-1)&1;
  cseymod2=1-cseymin1mod2;
  csexmod3x3= -3;
  csexx2mod160_1_2=cseymod2-8;
  csexx2div160_1_2=0;
  for (x=0; x<N+5; ++x) {
    csexmin1mod3x3=csexmod3x3;
    csexmod3 ++;
    if(csexmod3 >= 3){csexmod3 -= 3;}
    csexmod3x3=csexmod3*3;
    csexmin2mod3x3 = 9-csexmod3x3-csexmin1mod3x3;
    csexx2mod160_1_2+=2;
    csexmin1x2 = (x-1)*2;
    csexmin3x2 = csexmin1x2-4;
    if(csexx2mod160_1_2>=160) {csexx2mod160_1_2-=160;csexx2div160_1_2++;}
                ...
    if (x>=3 && x<N+3 && y>=2 && y<M+2)
      tmparray[csexx2mod160_1_2 + csexx2div160_1_2*256 + 96]
        = comp_edge_pixels[csexmod3x3
                           + cseymin2mod3] = maxdiff_compute;
                ...
    if (x>= 1 && x<N+1 && y>=1 && y<=M)
      tmparray[((csexmin1x2 + cseymin1mod2)&63)
                        + ((csexmin1x2 + cseymin1mod2)>>6)*256]
        = gauss_xy_pixels[csexmin1mod3x3
                           + cseymin1mod3] = gauss_xy_compute;

    ...
  }
}
```

FIG. 12

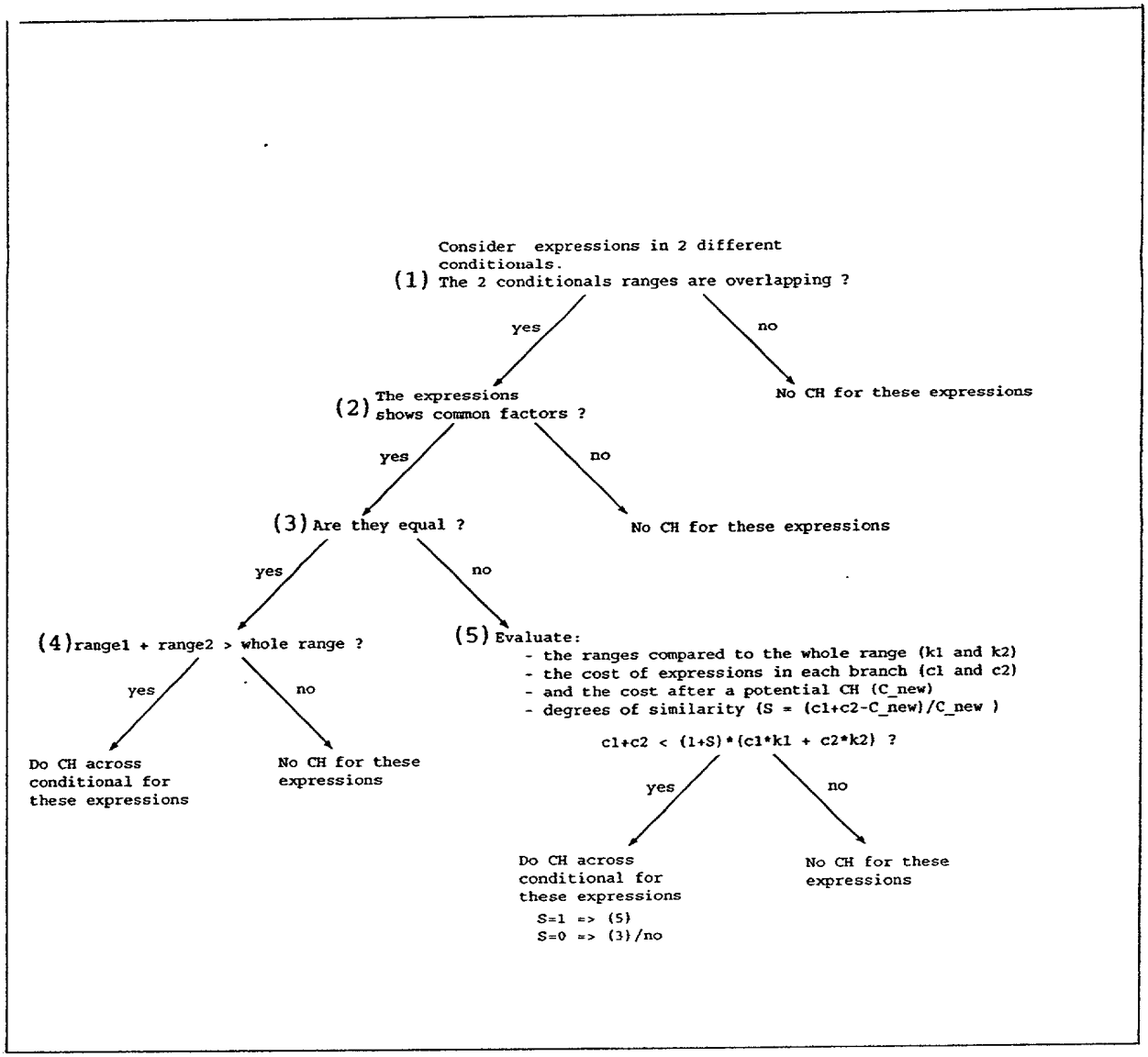
Consider expressions in 2 different conditionals.
(1) The 2 conditionals ranges are overlapping ?
yes
no
No CH for these expressions
(2) The expressions shows common factors ?
yes
no
No CH for these expressions
(3) Are they equal ?
yes
no
(4) range1 + range2 > whole range ?
yes
no
Do CH across conditional for these expressions
No CH for these expressions
(5) Evaluate:
- the ranges compared to the whole range (k1 and k2)
- the cost of expressions in each branch (c1 and c2)
- and the cost after a potential CH (C_new)
- degrees of similarity (S = (c1+c2-C_new)/C_new)
c1+c2 < (1+S)*(c1*k1 + c2*k2) ?
yes
no
Do CH across conditional for these expressions
S=1 => (5)
S=0 => (3)/no
No CH for these expressions

FIG. 13